Amendment to the Claims:

This listing of claims will replace all versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A system for object-to-object mapping between an operating environment and native processor language comprising:

a memory, the memory including a first memory portion storing an application inclusive of at least one application side object description, which at least one object description is compatible computer language associated with the application, wherein each application side object includes associated application side identification data;

a second memory portion of the memory, the second memory portion storing executable code in a format native to a selected processor, which executable code includes at least one native side object description, wherein each native side object description includes associated native side identification data;

an object map means adapted for mapping between the at least one application side object description and the native side object description, the object map including means adapted for associating application side identification data with native side identification data;

a class identifier for identifying a class associated with the at least one application side object description and the at least one native side object description;

testing means adapted for testing the object map to determine whether a particular pointer is associated with a key data component;

mapping means for mapping between a selected application side object description and a selected native side object description in accordance with an output of the testing means;

means adapted for generating updated key data in accordance with an output of the mapping means; ~~and~~

a class loader for loading a native class factory for instantiation of objects associated with an identified class; and

means adapted for completing access to an object via the object map without translation between the application side and the native side.

2. (Original) The system for object-to-object mapping between an operating environment and native processor language of Claim 1, further comprising:

means adapted for determining a presence of the at least one application side object; and

means adapted for initiating the class loader in accordance with a determined presence of the at least one application side object.

3. (Original) The system for object-to-object mapping between an operating environment and native processor language of Claim 2, further comprising:

class factory testing means adapted for testing whether a class factory is still required after a load thereof; and

means adapted for selectively unloading a native class factory in accordance with an output of the class factory testing means.

4. (Original) The system for object-to-object mapping between an operating environment and native processor language of Claim 2, further comprising:

object testing means adapted for testing whether a native side object description is out of a pre-selected scope; and

garbage collection means adapted for removing references to a native side object in accordance with an output of the object testing means.

5. (Original) The system for object-to-object mapping between an operating environment and native processor language of Claim 4 further comprising means adapted for periodically enabling operation of the garbage collection means.

6. (Original) The system for object-to-object mapping between an operating environment and native processor language of Claim 5 wherein the means for periodically enabling operation of the garbage collection means is periodically enabled on a pre-selected iteration.

7. (Currently Amended) The system for object-to-object mapping between an operating environment and native processor language of Claim 3 wherein the associated native side identification data includes an object pointer~~, and wherein the associated application side identification data a jobject of a JAVA native interface~~.

8. (Currently Amended) A method for object-to-object mapping between an operating environment and native processor language comprising the steps of:

storing, in a first memory portion of a memory, an application inclusive of at least one application side object description, which at least one object description is compatible computer language associated with the application, wherein each application side object includes associated application side identification data;

storing, in a second memory portion of the memory, executable code in a format native to a selected processor, which executable code includes at least one native side object description, wherein each native side object description includes associated native side identification data;

mapping, by means of an object map, between the at least one application side object description and the native side object description, the object map including means adapted for associating application side identification data with native side identification data;

identifying a class associated with the at least one application side object description and the at least one native side object description; ~~and~~

testing the object map to determine whether a particular pointer is associated with a key data component;

mapping between a selected application side object description and a selected native side object description in accordance with an output of the testing means;

generating updated key data in accordance with an output of the mapping means;

loading a native class factory for instantiation of objects associated with an identified class; and

completing access to an object via the object map without translation between the application side and the native side.

9. (Original) The method for object-to-object mapping between an operating environment and native processor language of Claim 8, further comprising the steps of:

determining a presence of the at least one application side object; and

initiating the class loader in accordance with a determined presence of the at least one application side object.

10. (Original) The method for object-to-object mapping between an operating environment and native processor language of Claim 9, further comprising the steps of:

testing whether a class factory is still required after a load thereof; and

selectively unloading a native class factory in accordance with an output of the testing.

11. (Original) The method for object-to-object mapping between an operating environment and native processor language of Claim 9, further comprising the steps of:

testing whether a native side object description is out of a pre-selected scope; and

removing, by means of a garbage collection means, references to a native side object in accordance with an output of the testing.

12. (Original) The method for object-to-object mapping between an operating environment and native processor language of Claim 11 further comprising the step of periodically enabling operation of the garbage collection means.

13. (Original) The method for object-to-object mapping between an operating environment and native processor language of Claim 12 wherein the step of periodically enabling operation of the garbage collection means is periodically enabled on a pre-selected iteration.

14. (Currently Amended) The method for object-to-object mapping between an operating environment and native processor language of Claim 10 wherein the associated native side identification data includes an object pointer~~, and wherein the associated application side identification data a jobject of a JAVA native interface~~.

15. (Currently Amended) A computer readable medium of instructions for object-to-object mapping between an operating environment and native processor language comprising:

a memory, the memory including of first memory portion storing an application inclusive of at least one application side object description, which at least one object description is compatible computer language associated with the application, wherein each application side object includes associated application side identification data;

a second memory portion of the memory, the second memory portion storing executable code in a format native to a selected processor, which executable code includes at least one native side object description, wherein each native side object description includes associated native side identification data;

an object map means adapted for mapping between the at least one application side object description and the native side object description, the object map including means adapted for associating application side identification data with native side identification data;

a class identifier for identifying a class associated with the at least one application side object description and the at least one native side object description; ~~and~~

testing means adapted for testing the object map to determine whether a particular pointer is associated with a key data component;

mapping means adapted for mapping between a selected application side object description and a selected native side object description in accordance with an output of the testing means;

means adapted for generating updated key data in accordance with an output of the mapping means;

a class loader for loading a native class factory for instantiation of objects associated with an identified class; and

means adapted for completing access to an object via the object map without translation between the application side and the native side.

16. (Original) The computer readable medium of instructions for object-to-object mapping between an operating environment and native processor language of Claim 15, further comprising:

means adapted for determining a presence of the at least one application side object; and

means adapted for initiating the class loader in accordance with a determined presence of the at least one application side object.

17. (Original) The computer readable medium of instructions for object-to-object mapping between an operating environment and native processor language of Claim 16, further comprising:

class factory testing means adapted for testing whether a class factory is still required after a load thereof; and

means adapted for selectively unloading a native class factory in accordance with an output of the class factory testing means.

18. (Original) The computer readable medium of instructions for object-to-object mapping between an operating environment and native processor language of Claim 16, further comprising:

object testing means adapted for testing whether a native side object description is out of a pre-selected scope; and

garbage collection means adapted for removing references to a native side object in accordance with an output of the object testing means.

19. (Original) The computer readable medium of instructions for object-to-object mapping between an operating environment and native processor language of Claim 18 further comprising means adapted for periodically enabling operation of the garbage collection means.

20. (Original) The computer readable medium of instructions for object-to-object mapping between an operating environment and native processor language of Claim 19 wherein the means for periodically enabling operation of the garbage collection means is periodically enabled on a pre-selected iteration.

21. (Currently Amended) The computer readable medium of instructions for object-to-object mapping between an operating environment and native processor language of Claim 17

wherein the associated native side identification data includes an object pointer~~, and wherein the associated application side identification data a jobject of a JAVA native interface~~.

22. (Currently Amended) A computer implemented method for object-to-object mapping between an operating environment and native processor language comprising the steps of:

storing, in a first memory portion of a memory, an application inclusive of at least one application side object description, which at least one object description is compatible computer language associated with the application, wherein each application side object includes associated application side identification data;

storing, in a second memory portion of the memory, executable code in a format native to a selected processor, which executable code includes at least one native side object description, wherein each native side object description includes associated native side identification data;

mapping, by means of an object map, between the at least one application side object description and the native side object description, the object map including means adapted for associating application side identification data with native side identification data;

identifying a class associated with the at least one application side object description and the at least one native side object description; ~~and~~

testing the object map to determine whether a particular pointer is associated with a key data component;

mapping between a selected application side object description and a selected native side object description in accordance with an output of the testing means;

generating updated key data in accordance with an output of the mapping means;

loading a native class factory for instantiation of objects associated with an identified class; and

completing access to an object via the object map without translation between the application side and the native side.

23. (Original) The computer implemented method for object-to-object mapping between an operating environment and native processor language of Claim 22, further comprising the steps of:

determining a presence of the at least one application side object; and

initiating the class loader in accordance with a determined presence of the at least one application side object.

24. (Original) The computer implemented method for object-to-object mapping between an operating environment and native processor language of Claim 23, further comprising the steps of:

testing whether a class factory is still required after a load thereof; and

selectively unloading a native class factory in accordance with an output of the testing.

25. (Original) The computer implemented method for object-to-object mapping between an operating environment and native processor language of Claim 23, further comprising the steps of:

testing whether a native side object description is out of a pre-selected scope; and

removing, by means of a garbage collection means, references to a native side object in accordance with an output of the testing.

26. (Original) The computer implemented method for object-to-object mapping between an operating environment and native processor language of Claim 25 further comprising the step of periodically enabling operation of the garbage collection means.

27. (Original) The computer implemented method for object-to-object mapping between an operating environment and native processor language of Claim 26 wherein the step of periodically enabling operation of the garbage collection means is periodically enabled on a pre-selected iteration.

28. (Currently Amended) The computer implemented method for object-to-object mapping between an operating environment and native processor language of Claim 24 wherein

the associated native side identification data includes an object pointer~~, and wherein the associated application side identification data a jobject of a JAVA native interface~~.